VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2019 František Krupa

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **František Krupa**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: ITA spol. s r.o. Ostrava
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Radecký, Ph.D.**

Konzultant bakalářské práce: Ing. Pavel Šimeček, Ph.D.

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Darkovicích 24. dubna 2019 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 24. dubna 2019

.....

Děkuji všem, kteří mi s touto bakalářskou prací pomohli. Především Ing. Pavlovi Šimečkovi, Ph.D. a Ing. Danielovi Hajdukovi, Ph.D., kteří mi poskytli možnost pracovat v jejich firmě ITA spol. s.r.o. a vedoucímu mé bakalářské praxe Ing. Michalovi Radeckému, Ph.D.

Abstrakt

Tato práce pojednává o mé individuální odborné praxi ve firmě ITA spol. s.r.o. na pozici C++ programátora v Ostravě.

Součástí tohoto dokumentu je popis firmy, mé pracovní pozice, programů, na nichž pracuji, a zadané úkoly s jejich následným řešením.

V závěru práce jsou uvedené uplatněné a scházející praktické znalosti, krátký popis průběhu praxe a popis užitečných vysokoškolských předmětů k praxi.

Klíčová slova: ITA spol. s r.o., Individuální odborná praxe, C++ MFC

Abstract

This work is about my individual professional practice in ITA spol. s.r.o. as C++ programmer in Ostrava.

This document includes description of company, my working position, programs, in which I work, and given tasks with their solution.

At the end of this work is stated used and missing practical knowledge, short description of my workflow and description of useful school subjects.

Key Words: ITA spol. s r. o., Individual Professional Practice, C++ MFC

Obsah

Seznam použitých zkratek a symbolů

MFC	–	Microsoft Foundation Classes
SVN	–	Subversion
IDE	–	Integrated Development Environment
CSV	–	Comma Separated Values
XML	–	Extensible Markup Language

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

1.1 Profil firmy

Společnost ITA spol. s.r.o. je soukromá česká společnost, založená v roce 1991 výzkumnými a vědeckými pracovníky Výzkumného ústavu strojírenského a metalurgického VÍTKOVICE. Zabývá se moderními technologiemi válcování za tepla i za studena, dodává know-how a programová řešení významným dodavatelům válcovacích zařízení, technologií a řídicích systémů jako např. Danieli, ConverTeam, Acos Vilares, Vítkovice Heavy Machinery. Dále řeší technické problémy a technologické inovace na teplých, či studených válcovacích tratích jako např. ArcelorMittal Ostrava, Severstal Čerepovec, ArcelorMittal Vanderbijlpark, CSN Voltaredunda, Třinecké železárny Třinec

Mezi hlavní aktivity patří řešení technických problémů, zpracování studií, konzultační a poradenská činnost v oblasti technologií válcování za tepla i za studena, vývoj nových, optimalizace a úpravy stávajících SW modulů řídicích systémů válcovacích tratí (Level2), vývoj speciálních programů určených pro off-line simulace procesů válcování a ochlazování, počítačové modelování procesů tváření a tepelného zpracování.

1.2 Pracovní pozice

Ve firmě ITA spol. s.r.o. jsem po přijímacím řízení nastoupil jakožto brigádník, na pozici programátor, až poté jsem si domluvil bakalářskou praxi. Obvykle pracuji ve skupině dvou, až čtyř lidí. Denně dostáváme informace o nutných úpravách v programu a práci si rozdělujeme po jednotlivých bodech. Snažíme se, pokud možno, řešit vše metodou per partes.

V této firmě jsem byl zaučen na programech QTSteel a DLPP, kde jsem začínal s malými úpravami a učil se vyhledávat ve zdrojových kódech. Především jsem začínal děláním úprav na programu QTSteel ve verzi VVT.

Později jsme vytvářeli 3 úplně nové verze v programu QTSteel, z čehož v době psaní této práce je již jedna hotová, druhá ve fázi finálního ladění a poslední je stále ve vývoji. Mou specializací je práce s firemní knihovnou ContourLib, což je knihovna psaná v C++ a OpenGL, která zajišťuje komplexní práci s 1D a 2D grafy. Tato knihovna je schopná zobrazit i 3D grafy, ale již bez mnohých funkcí.

2 Používané technologie

2.1 Používané jazyky

2.1.1 C++

C++ je programovací jazyk, jenž vznikl v roce 1985 a podporuje několik programovacích paradigmat. Především objektově orientované programování, procedurální programování a generické programování. Tento jazyk je nejvhodnější, pokud nám záleží především na výpočetní rychlosti a kontrole nad pamětí.

2.1.2 MFC

Microsoft Foundation Class Library, zkráceně MFC, je knihovna, jenž nám umožňuje používat Windows API a snadněji pracovat a vytvářet různá okna, či kontrolní panely. Tato knihovna je jednou z nejrozšířenějších a nejznámějších knihoven pro práci s grafickým rozhraním v OS Windows.

2.1.3 XML

Extensible Markup Language, zkráceně XML, je obecný značkovací jazyk. Je vhodný pro přehledné a editovatelné ukládání informací do souboru. Používá se především pro serializaci dat, či jinou práci se vstupními a výstupními daty.

2.2 Používané programy

2.2.1 Visual Studio 2012

Visual studio 2012 je IDE od firmy Microsoft. Používá se pro vytváření konzolových aplikací, aplikací s grafickým rozhraním, webových stránek, služeb, atd. Obsahuje editor kódu podporující IntelliSense (našeptávač, který napovídá existující metody, proměnné a jiné práci usnadňující informace), dále refaktoring (proces způsobující lepší čitelnost kódu) a debugger (utilita pro ladění a hledání chyb ve spuštěném programu)

2.2.2 TortoiseSVN

SVN je systém pro správu a verzování zdrojových kódů. TortoiseSVN je verzovací klient, implementovaný ve Windows kontextovém menu, který pomáhá k rychlému odeslání aktuálních zdrojových kódů na server a následnému spojení kódu od více programátorů, kteří tak mohou pracovat na jednom zdrojovém kódu současně. V mém případě se používá na denní sjednocení zdrojových kódů.

2.2.3 GitHub

GitHub je webová služba používající verzovací nástroj Git, což je nástroj, který umožňuje správu kódu, sledování změn a ukládání jednotlivých verzí programu. Ve firmě je používán především na ukládání verzí, které již putují k zákazníkovi. Slouží tedy k vyvolání stejné chyby, která se stala zákazníkovi, což nám umožňuje sjednat nápravu chyby.

2.2.4 Inno Setup

Inno Setup je program sloužící k vytvoření instalačních a odinstalačních průvodců. Podporuje všechny OS Windows, od Windows 2000, až po Windows 10. Program umožňuje instalované soubory zkomprimovat, spouštět jiné programy před, během, či po instalaci, vytvářet registry, atd.

2.3 Firemní programy

2.3.1 Program QTSteel

Program QTSteel je určen pro výpočet strukturních podílů a mechanických vlastností ocelí po tepelném zpracování. Umožňuje také zobrazení rozpadového diagramu oceli daného chemického složení, výpočet ochlazovacích křivek speciálních těles (tyč kruhového nebo obdélníkového průřezu, válec, prstenec, trubka) pro uživatelem zadanou posloupnost ochlazovacích podmínek (databáze ochlazovacích médií). Podporuje také vložení teplot obecných 2D-těles vypočítaných v MKP programech (FLUX, FormFEM). Dále obsahuje výpočet obsahu strukturních složek (procenta feritu, perlitu, bainitu a martenzitu) a mechanických vlastností oceli (tvrdost HV, HB, HRC, mez kluzu, mez pevnosti v tahu) po kalení a následném popuštění/žíhání po průřezu tepelně zpracovávaného tělesa nebo v jeho bodech pod povrchem a výpočet průběhu mechanických vlastností po kalení (popuštění/žíhání) v závislosti na hloubce pod povrchem tepelně zpracovávaného tělesa.

2.3.2 Program DLPP

DLPP neboli Danieli Long Products Properties Predictor byl vytvořen roku 2005 společností ITA spol. s.r.o. Byl vyvinut pro off-line počítačovou simulaci metalurgických procesů při válcování za tepla, nebo po následném ochlazení, válcovaných tyčí a drátů.

Na základě specifikovaných chemických vlastností oceli a dle technologie válcování, program předpovídá parametry mikrostruktury deformovaného austenitu po válcování.

3 Popis zadaných úkolů

3.1 Program QTSteel

3.1.1 Zaučení se s programem

Mé první úkoly ve firmě byly, vyhledat proměnné pro textová pole v určeném dialogu a ošetřit jejich povolený rozsah, ať už počet znaků, nebo číselný rozsah. Dále zaokrouhlit čísla na určitý počet desetinných míst, přidat tlačítka, combo boxy, radio buttony a napojit na ně metody. Poté upravování defaultních hodnot, stávajících funkcí a metod, rozšiřování serializace na nové třídy, struktury a proměnné, načítání souborů XML, či CSV a vytváření logů.

3.1.2 Verze *PACK*

Česká subverze programu QTSteel vytvořená pro firmu ArcelorMittal Ostrava a.s. Tato subverze má počítat s pěti předdefinovanými 2D profily, kdy největší překážkou v této verzi je, že tato tělesa jsou na určitých sekcích složena vedle sebe, v "packu", a předávají si tak navzájem teplo. Program QTSteel není stavěný na výpočet několika těles současně a musel se tedy vytvořit nový algoritmus simulace a upravit knihovnu pro simulaci ochlazování 2D těles zvaná TempFEM2D.

1. Etapa vývoje programu:

- Přidat novou definici verze s pracovním názvem *PACK* do customizace.
- Vytvořit úvodní dialog po spuštění programu před zobrazením hlavního okna programu, kde se bude vybírat, zdali bude aktivní "pack annealing"(skládání těles). Dále se v tomto dialogu bude vybírat typ profilu, který už dále v programu nepůjde změnit. Taktéž se bude vybírat typ ochlazovacího úseku a možnosti vytvořit nový projekt, nebo otevřít již existující projekt.
- Upravení vlastností ocelí v závislosti na použitém typu profilu a s tím související úprava chemické tabulky.
- Načtení ochlazovací linky ze souboru XML, podle zvoleného typu.
- Načtení profilu, podle zadaného typu a rozměrů.
- Uvedení do funkčního stavu základní ochlazovací simulaci bez aktivního skládání těles.

2. Etapa vývoje programu:

- Vytvořit algoritmus pro vyhledání jednotlivých hran obrysu profilu, na něž se budou přiřazovat ochlazovací podmínky.

- Vytvořit jednotlivé ochlazovací strany (soubor hran), které se mění v závislosti na zadané velikosti profilu. Vytvořit třídu, která přidělí ochlazovací podmínky, podle aktuálně simulované sekce, na ochlazovací strany.
- Úprava stávajícího algoritmu pro chladící simulaci, která bude schopná měnit ochlazovací podmínky za běhu a měnit hodnoty aktuální ochlazovací teploty na teplotu vedlejšího tělesa.

3. Etapa vývoje programu:

- Na 2D grafu výsledku simulace umožnit výběr myší bod na profilu, který vytvoří 1D teplotní ochlazovací křivku v závislosti čase.
- Nalézt a odstranit memory leaky, jelikož program kvůli přetečení maximální povolené velikosti na paměti spadne.

3.1.3 Verze *DLPPS*

Za úkol bylo zde vytvořit novou anglickou subverzi programu QTSteel, pro italskou firmu, s předpokladem možného překladu do italštiny. Tato verze místo výpočtů předdefinovaných 2D a 1D tvarů, načte obecný 2D profil z výstupních souborů z externího programu Abaqus. Následně bylo úkolem provést metalurgické výpočty na jednotlivých průchodech a na profilu posledního průchodu provést uživatelem definovanou simulaci ochlazování.

1. Etapa vývoje programu:

- Přidat novou definici *DLPPS*.
- Upravit tabulku ochlazovacích podmínek tak, aby uživatel byl schopen si vytvářet a editovat jednotlivé podmínky a sdružovat je do sekcí.
- Vytvoření dialogu pro načtení výstupních Abaqus dat, pro jejich kontrolu jak vizuální, tak strukturální. Následně konvertovat tato data do vlastních struktur a serializovat je do souboru. Jednotlivé profily se budou zobrazovat pomocí knihovny ContourLib.
- Načíst profil posledního průchodu do simulace ochlazovaní a upravit simulaci pro obecné 2D těleso.
- Vytvořit úvodní dialog, zobrazovaný před spuštěním hlavního okna programu, ve kterém bude možné otevřít si již načtená a konvertovaná data z programu Abaqus. Dále umožnit vytvoření jak nového projektu, tak načtení již existujícího serializovaného projektu verze *DLPPS*.

2. Etapa vývoje programu:

- Vytvořit modální a nemodální dialog, kdy v nemodálním dialogu budou zobrazeny jednotlivé ochlazovací podmínky z určité sekce, ze kterých půjde vybírat a každá bude mít přiřazenou barvu. V modálním dialogu bude zobrazena kostra profilu a zvýrazněný její obrys. Na jednotlivé hrany bude pomocí myši, možné přiřazovat jednotlivé ochlazovací podmínky, které se poté využijí při ochlazovací simulaci. Umožnit vytvoření globálních makro hran na určitý profil, které půjde opět vybrat myší.
- Umožnění uživateli vytvořit si více technologií ochlazování.
- Umožnit vytváření 1D ochlazovacích křivek v závislosti na čase výběrem bodu na vypočteném profilu jako v *PACK* verzi. (V čase zadání tohoto úkolu byl již vyřešen stejný úkol v *PACK* verzi.) Dále tuto funkci rozšířit o přichytávání bodu k nejbližšímu okraji, pokud je ukazatel myši v uživatelem definované vzdálenosti od okraje. Tuto rozšířenou funkci dále implementovat do všech stávajících subverzí programu, využívající vytváření křivek, podle vybraného bodu.

Další Etapa vývoje tohoto programu je ve vývoji a tedy nemám dostupné informace, co bude dále následovat.

3.1.4 Verze *BTMP*

Verze *BTMP* je anglická verze, s možným budoucím překladem do polštiny. Ze zmíněných verzí vzniká jako poslední. Jedná se o verzi mající 2 profily z *PACK* verze, s mírně upravenou bodovou sítí. Obsahuje také základní ochlazovací simulaci s možností vytvořit si vlastní sekce jako v *DLPPS*, avšak bez možnosti přidat ochlazovací podmínky na určité části tělesa. Zobrazování výsledků má fungovat jako v *DLPPS* a má také podporovat výběr bodů, i s přichytáváním k okraji, pro vytvoření ochlazovacích křivek.
Seznam zadaných úkolů:

- Vytvoření nové definice *BTMP*.
- Vytvoření dialogu pro spuštění programu, podobně jako ve verzi PACK, rozšířenou o nastavení jakosti oceli.
- Načíst profily, nalézt hrany okraje, přiřadit neměnné ochlazovací strany a upravit simulaci, aby byla spustitelná.
- Výsledky zobrazit jako v *DLPPS* verzi.

3.2 Program DLPP

3.2.1 Předklad do italštiny

Celý stávající program DLPP a všechny jeho knihovny přeložit do Italštiny.

3.2.2 Čínská verze

Vytvořit novou verzi instalace čínské verze a následně zkontrolovat funkčnost s překladem.

3.2.3 Opravy

Během mé praxe se podílet i na různých malých opravách a přidávání funkcionalit do programu DLPP.

4 Řešení zadaných úkolů

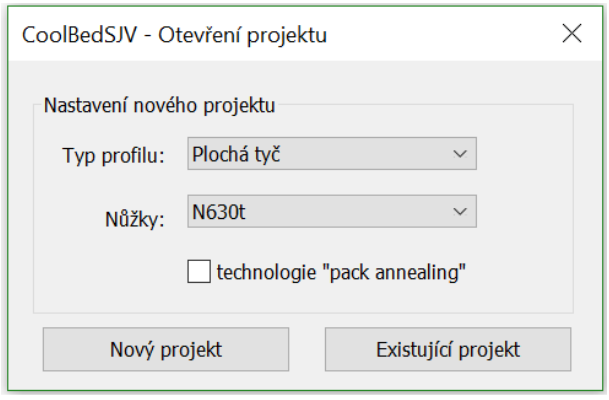
4.1 Verze *PACK*

4.1.1 Přidání definice verze

Ze začátku jsme přidali definice nového zákazníka do patřičných tříd. Jelikož se verze mění pomocí preprocesoru ve vlastnostech projektu a z důvodu častého opravování i jiných verzí, by bylo pracné stále přepisovat direktivy v preprocesoru, přidal jsem následně do visual studia novou konfiguraci, která tento problém vyřešila. Pak jsme vytvořili nové a editovali kopie stávajících dialogů. Dále jsme vytvořili kopie těch tříd, jenž se význačně měnily, abychom nedopatřením nezměnili funkčnost i v ostatních verzích programu. Pokud však bude nutno změnit funkčnost globálně, pro určitou metodu, bude zapotřebí provést změny ve všech kopiích určité třídy. Často se tak volí mezi jednodušší editovatelností s lepší čitelností kódu a časově náročnější editací globálně užívaných metod.

4.1.2 Vytvoření spouštěcího dialogu

Následovalo vytvoření dialogu, pro spuštění a vybrání simulovaného tělesa a linky. Některá tělesa ale měla určitá pravidla, dle kterých se musel řídit i combo box s nastavením linky a check box aktivního skládání těles. Vznikla tedy metoda, která tato jednotlivá nastavení měnila v závislosti na tělese, primárně pomocí funkce GetDlgItem. Při ukončení dialogu, ať už vytvořením nového projektu, nebo načtením již existujícího, byl problém přenést nastavená data do další části programu, jelikož hlavní objekt, ze kterého čerpá celé hlavní okno, ještě nebyl inicializován. Když jsme tedy posunuli kód pro spouštění dialogu, zobrazilo se hlavní okno, které ještě nemělo být zobrazeno. Vytvořili jsme tedy pomocnou globální třídu, pro ukládání dat pro spuštění, která je dodnes hojně využívaná i v ostatních verzích programu.



Obrázek 1: Ukázka spouštěcího dialogu pro verzi *PACK*

4.1.3 Úprava dialogu "Vlastnosti oceli"

Dalším úkolem byla úprava dialogu "Vlastnosti oceli", kde v kódu programu byly jednotlivé oceli označovány čísly. V customizaci jsem definoval čtyři oceli s označením 17 a jednu ocel s označením 31. Po inicializaci programu byly však v chemickém poli pouze 2 chemie z 5 definovaných. Chyba tedy byla, že inicializace chemie vytvořila pole se všemi existujícími chemiemi a následně mazala ty chemie, které nebyly definovány v customizaci. Nebylo tedy možné mít více chemií stejného označení. Opravil jsem tedy inicializaci chemie, aby se vytvářely pouze definované customizací a následně se tak nic nemuselo mazat. Při editování třídy, pro tento dialog, jsme psali kód do již existující třídy a nevytvořili její kopii. Níže je ukázka podmíněného překladu pro verzi *PACK*, která se musela použít vždy, když úprava byla jen pro tuto verzi. Kód se tak po větších úpravách stal hůře čitelným.

```
#ifdef MODE_PACK
   // Code for PACK
#else
   // Code for other versions
#endif
```

Výpis 1: Užití prodmíněného překladu pro verzi *PACK*

4.1.4 Načtení a optimalizace linky

Dle zvolené linky, ve vstupním dialogu, jsem načetl patřičný XML soubor. Během serializace XML, se v průběhu čtení vytváří ochlazovací sekce (dále jen sekce) a na nich ochlazovací podmínky (dále jen podmínky). První problém, který se u vytváření sekcí a podmínek vyskytl, je, že objekt podmínky se neukládá do pole objektu sekce, ale pole sekcí se ukládá v podmínce. Z toho vyplývá, že jsem musel předem definovat, kolik celkem podmínek bude na jednotlivých sekcích a až poté do všech podmínek ukládat sekce. Z tohoto vznikl další problém, kdy bylo zapotřebí vytvořit N počet podmínek s M počtem typů podmínek, kam by se ukládal L počet sekcí, z čehož vznikl N M L součin objektů sekcí. Toto bylo s větším počtem sekcí a podmínek redundantní a pro debug účely nepřehledné. Proto jsem vytvořil pomocnou strukturu, která pomohla omezit pole podmínek, na maximální možný počet podmínek, na daném tělese, v jeden moment, což vytvářelo součin N L objektů sekcí. Ve verzi *PACK* je obvykle 9 sekcí s maximálně 5 podmínkami a 5 typy podmínek, což by bez pomocné struktury mohlo udělat až 225 objektů se sekcemi, místo aktuálních 45 objektů. V pomocné struktuře je uložen název podmínky, celočíselné označení typu, index sekce a index podmínky.

4.1.5 Načtení profilu a jeho škálování

Následně bylo potřeba načíst profil simulovaného tvaru, který byl vytvořen ve firemním programu FormFEM. Načtení binárního souboru bylo již vytvořeno, v původní verzi QTSteelu pomocí serializace, a tak se jen načítané profily lišily podle zadaných rozměrů, neboť při větších rozměrech se používaly profily s větší hustotou bodů, pro přesnější výsledek simulace. Jednotlivé soubory tak měly rozsahy, v kterých se používaly, načež bylo potřeba bodovou síť škálovat do požadovaných rozměrů. Bylo tedy zapotřebí zjistit rozměry načteného tělesa, což jsem vypočetl součtem absolutních maximálních souřadnic bodů, zvlášť na vertikální a horizontální ose. Poté jsem jednotlivé souřadnice bodů vynásobil vypočteným koeficientem pro zvětšení. Koeficient se počítá jako požadovaný rozměr, dělený rozměrem načteného tělesa. Dále jsem vytvořil konverzi načtených dat do sktruktur pro knihovnu ContourLib, která zobrazuje načtený profil.

4.1.6 Nalezení vnějších hran profilu

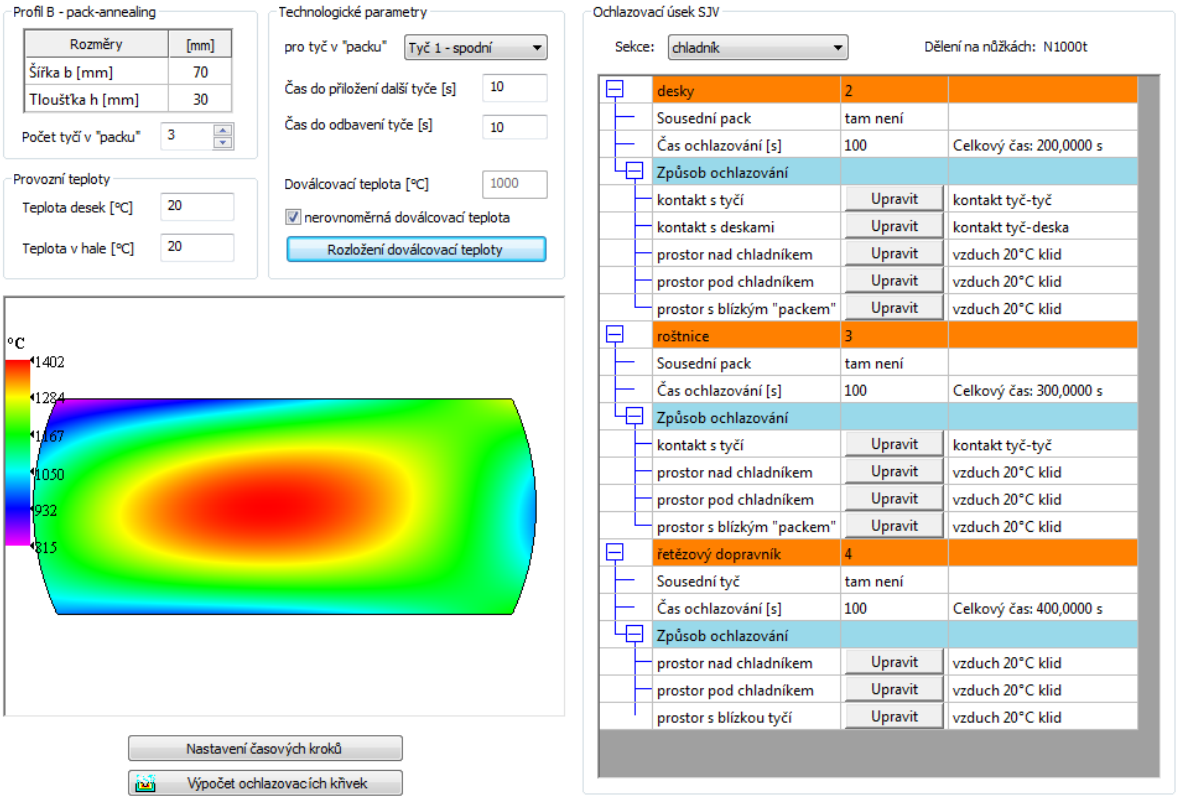
Dalším úkolem bylo najít hrany obrysu profilu. Jednotlivé body jsou spojeny v čtyřúhelníkové buňky, kdy každá hrana v buňce má svůj lokální index. Jednotlivé indexy hran buněk na obrysu je potřeba znát pro nastavení ochlazovacích podmínek. Pro nalezení bodů na obrysu bylo tedy možné použít sumu, kolikrát se určitý bod nachází ve všech buňkách. Ovšem v moment vytváření třídy, která měla sloužit pro stejný účel i ve verzi *DLPPS*, jsem použil jiného algoritmu z knihovny ContourLib, která dokázala počítat obrys na buňkách s jakýmkoliv počtem hran a zajistila se tak univerzálnost. Obrys je potřeba dále rozdělit na skupiny hran, k nimž se přiřazují ochlazovací podmínky. Body jsem tedy seřadil, aby šly po sobě, ve směru hodinových ručiček. Dále jsem tuto bodovou síť rozdělil do skupin, podle rozdílu po sobě jdoucích úhlu hran k normále horizontální osy. Měl jsem tedy zvlášť uložené pozice bodů, které byly v relativní rovině a které tvořily křivky, jenž nesměly změnit směr růstu úhlu. Tyto jednotlivé skupiny jsou ještě dále rozdělovány podle součtu délek hran.

```
void CPackOutline::CreateMacroPlochaTycPA(tag_OUTLINE *dDataStore, int PosOfRod ){
   double PosunX = 0.069, PosunY, PomY;

   if (m_iNpack*m_dH-100.0 > 0 )
      PosunY = (m_iNpack*m_dH -100.0)/1000.0;

   PomY = dDataStore->pNodes->GetMeshNode(m_vstackMacrosNodesIDs[0][0])->GetY();
   for (unsigned int i = 0; i < m_vstackMacrosNodesIDs.size(); i++)
   {
      for (unsigned int j = 0; j < m_vstackMacrosNodesIDs[i].size()-1; j++)
      {
         double ActualY = 0;
         double ActualX = 0;
         switch (i){
         case 0:
            ActualY = dDataStore->pNodes->GetMeshNode(m_vstackMacrosNodesIDs[i][j])->
                GetY();
            if ((PomY)-PosunY <= (ActualY))
               m_vMacroIndex.push_back(2);
            else
               m_vMacroIndex.push_back(3);
            break;
         case 1:
            m_vMacroIndex.push_back(1);
            break;
         case 2:
            m_vMacroIndex.push_back(6);
            break;
         case 3:
            ActualX = dDataStore->pNodes->GetMeshNode(m_vstackMacrosNodesIDs[i][j])->
                GetX();
            if ((m_dB/1000)-PosunX <= (ActualX))
               m_vMacroIndex.push_back(4);
            else
               m_vMacroIndex.push_back(5);
            break;
         }
      }
   }
}
```

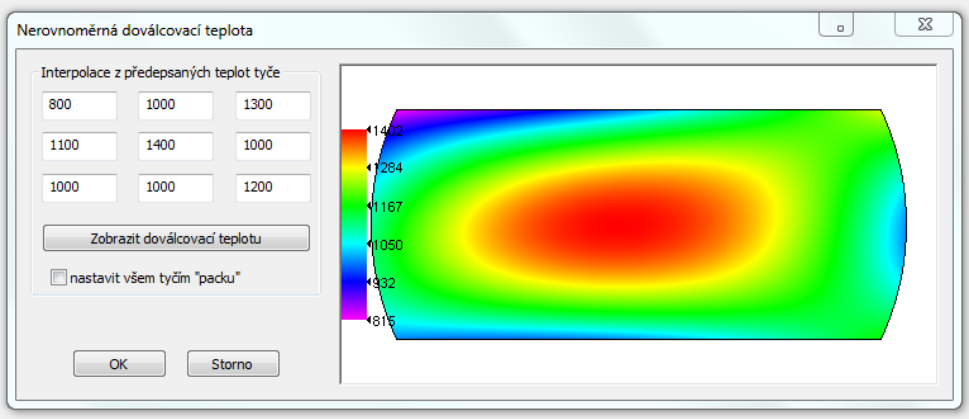
Výpis 2: Ukázka rozdělení hran, dle souřadnic, na ploché tyči s aktivním skládáním těles, pomocí již rozdělených 4 stran.

4.1.7 Teplotní simulace a interpolace teplot

Nyní jsem načtený profil konvertoval do struktur pro základní teplotní simulaci. Po úspěšné zkoušce simulace, jsme rozšířili vstupní a výstupní struktury a udělali z nich dynamická pole, pomocí datového kontejneru vector. Pro účely co nejpřesnější simulace jsem vytvořil dialog se zadáním teplot, pomocí devíti bodů, na tělese a ty se následně pomocí interpolace přenesou na celou bodovou síť. Následující krok je vypočíst jednotlivé pozice bodů. Jako prvním krokem bylo třeba najít v poli bodů nejkrajnější body, z čehož při součtu maxima a minima, s následným vydělením dvěmi, jsme nalezli středový bod. Od středového bodu se pomocí maxim a minim nalezly krajní body profilu, ležící na osách, a z těchto bodů se vytvořily ostatní body. Následně se použily funkce z interní knihovny TempFEM2D, pro interpolaci teplot na všechny body profilu.



Obrázek 2: Dialog nastavení ochlazovací simulace

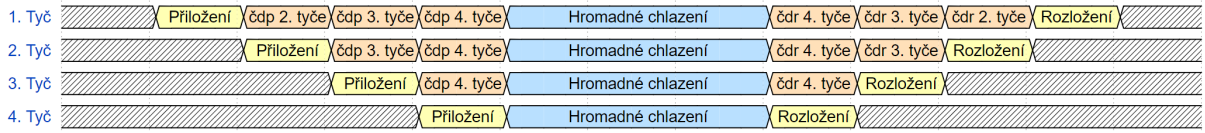


Obrázek 3: Dialog výpočtu nerovnoměrných teplot

4.1.8 Časový harmonogram simulace

Je zapotřebí pro každou tyč zvlášť vytvořit časový harmonogram, dle kterého by se řídila celá simulace. Cílem je tak vytvořit harmonogram, kdy již složené tyče musí mít stejné časové úseky, aby bylo možné provádět ochlazovací simulaci a předávání tepla mezi jednotlivými, již složenými tyčemi. Zde se vyskytla celá řada problémů. Rozdělili jsme tedy harmonogramy na 5 částí: před složením, skládání, složený "pack", rozkládání a po rozložení. Je také nutné rozlišovat, kdy a která tyč je v "packu"první, prostřední a poslední, aby tak bylo možné přiřazovat na tyto tyče podmínky ochlazování, a kdy určitou plochu ohřívá sousední tyč.

Jako pomyslný nultý čas jsme stanovili moment, kdy jsou všechny tyče složené. Od tohoto času, je pro jednotlivé tyče stanovený čas pro přiložení, kdy první tyč má tento čas roven nule, a všechny další tyče definoval uživatel, ale musel být větší než nula. Suma časů do přiložení následujících tyčí, se počítá jako čas navíc a tedy se musí připočíst k celkovému času, před pomyslným nultým časem, ke kterému se také připočte ochlazování před skládáním. Část se složeným "packem"nebylo třeba nijak ošetřovat, díky pomyslnému nultému času. Rozkládací část je algoritmus složení reverzně. Po rozložení už všechny tyče byly zvlášť, a tedy není problém s výpočtem zbytku časového harmonogramu. Nultý reálný čas, tak byla nejmenší hodnota a jeho absolutní hodnotu je nutno přičíst ke všem ostatním časům. Algoritmus pro simulaci byl tedy vytvořen na 2 části. První simuluje nesložené tyče a druhá simuluje tyče, jenž jsou v kontaktu s ostatními tyčemi. Přestup tepla mezi tyčemi byl prováděn nastavením obrysové teploty bodu jedné tyče, jako teplota ochlazovací podmínky nejbližšího bodu druhého tělesa a změnou koeficientu přestupu tepla na podmínce. Jak algoritmus pro harmonogramy, tak pro simulaci byl vícekrát přepisován, mnou i kolegou, jelikož se několikrát měnily podmínky skládání a rozkládání a často jsme nacházeli různé situace, kvůli kterým jsme museli algoritmy předělávat.



Obrázek 4: Obecný časový harmonogram skládání tyčí

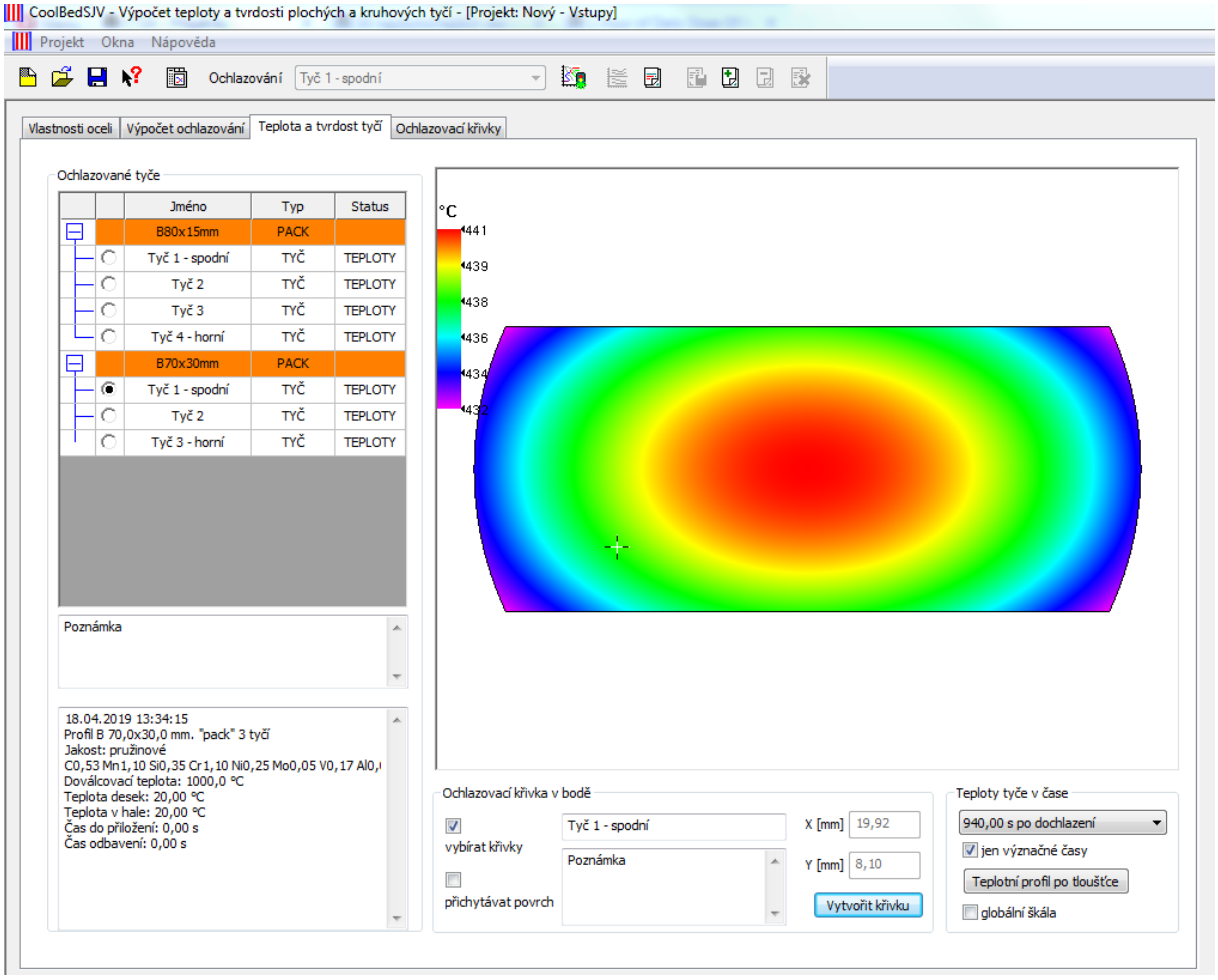
- *Šedě části - těleso se chladí samostatně, časy mohou být různé*
- *Žlutě části - těleso se přidává, nebo odkládá ze "zubu", na němž se tělesa chladí společně*
- *Oranžové části - tělesa se chladí společně a čekají na přiložení, nebo odebrání dalších tyčí (čdp - čas do přiložení, čdr - čas do rozložení)*
- *Modrá část - tělesa se chladí společně*
- *Všechny barevné časy musí mít společné časové kroky simulace*

4.1.9 Změna ochlazovacích podmínek během simulace

Simultánně se psaním algoritmu pro časový harmonogram a simulaci, jsem tvořil třídu, která se starala o měnění nastavení ochlazovacích podmínek na tělese. Třída obsahovala primárně přepínače, které měnily nastavení jednotlivých podmínek na hranách během simulace, podle aktuální simulované sekce a pozice tyče v "packu". K jednotlivým vytvořeným stranám byly přiřazeny předdefinované podmínky. Třída tak jen zjišťovala, ke které straně patří určitá hrana a přiřadila tak hraně, podle strany, podmínku.

4.1.10 Zobrazení výsledků simulace a vybrání bodu pro vytvoření ochlazovací křivky

Nyní bylo úkolem vytvořit rozšíření knihovny ContourLib a jejího manažeru, aby bylo možné vybrat na zobrazovaném grafu ochlazovaného tělesa bod, díky kterému se vytvoří ochlazovací teplotní křivka v čase. Vytvořil jsem tedy třídu, dědící z dosavadního manažeru knihovny ContourLib. V této třídě bylo třeba získat z knihovny pozici kliknutého bodu, která se musela přepočíst ze souřadnicového systému knihovny, do souřadnicového systému uložených výsledků. Vytvořil jsem tedy přepočet souřadnicových systémů. Z důvodu ukládání výsledků teplot na bodech, jsem vytvořil metodu, jenž nalezla buňku, ve které kliknutý bod leží. Z hodnot bodů buňky, se vypočte interpolovaná teplota v kliknutém bodě, ve všech časech. Během pozdějšího hlubšího poznání knihovny, při vytváření verze *DLPPS*, jsem nalezl metody, jenž přepočítávaly souřadnicový systém a také dokázaly nalézt buňku dle zadaných souřadnic. Vybraný bod na tělese je zobrazován jako nitkový kříž. Ten se však nevykresluje přímo na tělese, ale jen na okně s tělesem.



Obrázek 5: Zobrazení výsledku simulace s aktivním výběrem bodu pro vytváření ochlazovacích křivek

```
void CPackContourManagerEx::OnDraw(CDC *pDC, BOOL hideCross /* = FALSE*/)
{
    CContourManager::OnDraw(pDC);
    if (hideCross)
    {
        return;
    }
#if (PROGRAM_MODE == MODE_PACK)
    if (m_ImportPackdlg->GetbAllowAddingCurves() == TRUE)
#else
    if (m_ImportDlppsdlg->GetbAllowAddingCurves() == TRUE)
#endif
    {
        CWnd* pWnd = CWnd::FromHandle(FindFirstBody()->GetWndInfo()->GetHwnd());
        CDC* pDC = pWnd->GetDC();

        CPen *pOldPen, Pen;
        Pen.CreatePen(PS_SOLID, 0, RGB(0, 0, 0));
        pOldPen = pDC->SelectObject(&Pen);
        pDC->SetBkMode(TRANSPARENT);
        pDC->SetROP2(R2_XORPEN);

        CPoint pt = ContourSpace::GLToPt(*( FindFirstBody()->GetWndInfo()),
                                         ContourSpace::CDblPoint(m_pDoc->m_dXPoint, m_pDoc->
                                             m_dYPoint));

        int nSize = 10;

        pDC->MoveTo(pt.x - nSize, pt.y);
        pDC->LineTo(pt.x + nSize, pt.y);
        pDC->MoveTo(pt.x, pt.y + nSize);
        pDC->LineTo(pt.x, pt.y - nSize);
        pDC->SelectObject(pOldPen);
    }
}
```

Výpis 3: Vykreslení nitkového kříže na pozici zvoleného bodu, pomocí knihovny MFC, třídy CDC

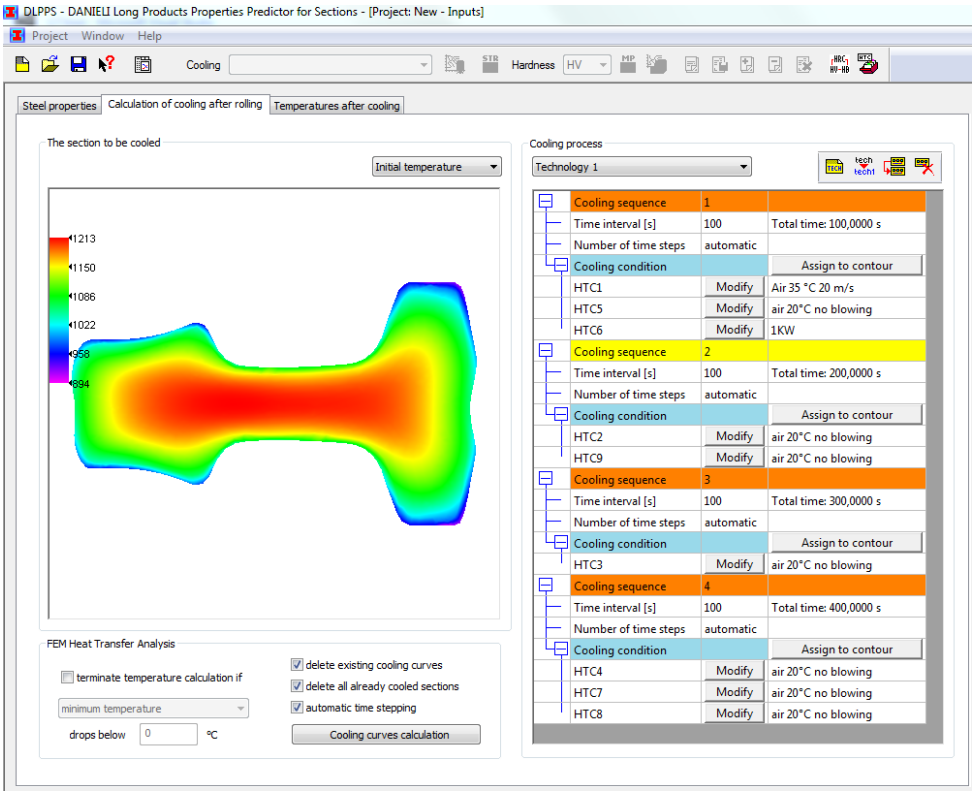
4.1.11 Optimalizace spotřeby paměti

Při testování této verze jsme zjistili, že během simulace vzniká spousty nedealokovaných objektů a počet bodů na profilu je přespříliš vysoký. Celý program tak po třech simulacích 4 tyčí, kdy každá měla 50 časových kroků, zabíral na paměti přes 2.5 GB dat a každá simulace přidávala přibližně 500-600 MB. Aby program nepadal, museli jsme v nastavení kompilátoru povolit zvětšený maximální alokovaný prostor. Po provedení optimalizace, opravení dealokování objektů a smazání již nepotřebných objektů jsme snížili spotřebu paměti při stejných podmínkách na přibližných 350 MB, kdy jedna simulace přidávala 30-50 MB. Nedealokované objekty jsme hledali pomocí logu, který vypisoval využití paměti. Dokázali jsme tak lokalizovat, která část programu nejvíce spotřebovává paměť a následně jí neuvolňuje. Všechny alokované objekty jsme přidali do dynamického pole a po ukončení části programu, kdy již bylo zřejmé, že tyto objekty nebudou zapotřebí, jsme je dealokovali. Vytvořili jsme tedy obdobu chytrých ukazatelů.

4.2 Verze *DLPPS*

4.2.1 Přidání nové definice verze

Přidali jsme novou definici verze *DLPPS* do programu, což znamená vytvoření ikony, definování aktivních typů ocelí, vytvoření defaultních dialogů a dalších nutnosti pro spuštění programu. Prvním úkolem bylo upravit tabulku s ochlazovacími podmínkami, aby bylo možné cokoliv editovat. Jak jsem již zmiňoval ve verzi *PACK*, jednotlivé sekce se ukládají v podmínkách, a tedy bylo potřeba vymyslet, jak uživateli umožnit vytváření podmínek a sekcí, bez omezení. Do třídy s podmínkou jsem přidal proměnnou, která značila, pro kterou sekci je tato podmínka vytvořena a pro ostatní sekce jsem tedy deaktivoval zobrazování této podmínky. Když uživatel vytvoří sekci, musí se do všech stávajících podmínek vložit nová sekce a vytvořit jedna nová podmínka, která má přiřazenou tuto sekci a je tedy v ní viditelná. Dle požadavků jsem také implementoval možnosti pro přidání sekcí na určitou pozici, posouvání sekcí v seznamu, mazání a kopírování sekcí, či podmínek. Upravil jsem také metodu, která vrací indexy aktuální zvolené podmínky a sekce. Původně byla navržena tak, že počítala indexy ke zvolenému řádku, dle celkového počtu sekcí a podmínek, avšak nyní se počet podmínek lišil dle toho, kolik podmínek je zobrazováno.



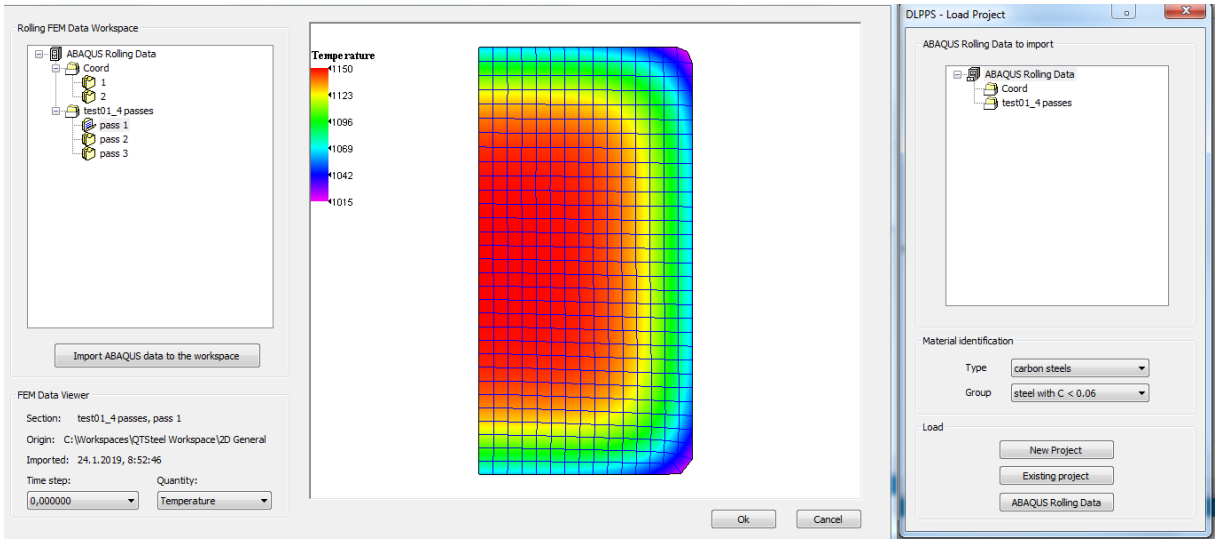
Obrázek 6: Nastavení ochlazovací simulace, již načteného profilu

4.2.2 Načtení souborů z externího programu

Následovalo vytvoření dialogu, který vytváří pomocí rekurzivní metody, strom složek a souborů. Po výběru složky, se načte celý profil s uloženými hodnotami. Když jsme vytvářeli načtení souborů, bylo zapotřebí pochopit, jak jsou soubory uloženy, neboť jsme dostali soubory s textovými daty bez jakéhokoliv popisu dat. Pomocí programu LibreOffice Calc jsme vytvářeli úseky profilu, abychom pochopili strukturu souborů. Po načtení složky s jednotlivými kroky si uživatel může zobrazit profily s hodnotami v čase. Vytvořili jsme také kontroly, zdali soubory na sebe navazují, jelikož se stávalo, že jsme měli profilovou síť 15. kroků, ale hodnotová data končila u 14. kroku. V tomto dialogu se v budoucnu budou počítat metalurgické výpočty změny tvrdosti a podobně. Načítání těchto vstupních dat, při každém spuštění programu, bylo příliš zdlouhavé a repetitivní. Proto jsme vytvořili ukládání do naších struktur, pomocí kterých je možné dále vytvářet nové projekty, pro program QTSteel, a načítání probíhá téměř okamžitě. Díky tomu, že jsme si uložili hodnoty do vlastních struktur, nebylo tedy třeba upravovat simulaci a program již počítal základní simulaci s jednou obecnou podmínkou pro celé těleso. Během simulace však docházelo k chybě, kdy knihovna TempFEM2D vstupní data profilu tělesa optimalizovala pro rychlejší výpočet a na konci výpočtu opět data převedla zpět do původní podoby. Výsledky simulace se totiž ukládaly pro optimalizované hodnoty, ale již se nepřeváděly na neoptimalizované. Byly tedy dvě možnosti. Vytvořit konverzi výsledků, což prodlužovalo čas potřebný pro simulaci, nebo nepřevádět zpět data. Do programu jsme přidali obě možnosti. Pro debug se používá rychlejší varianta a pro release pomalejší, kdy v release verzi není zdržení simulace znatelné, avšak je větší jistota správnosti výsledku. Tato chyba nikdy předtím nebyla objevena, jelikož program FormFEM, ze kterého byly všechny doposud simulované profily, již optimalizoval sítě, a tedy optimalizace proběhla beze změny dat.

4.2.3 Vytvoření spouštěcího dialogu

Dalším úkolem bylo vytvoření dialogu před spuštěním hlavního okna, pro výběr uložených konvertovaných profilů, typu oceli, či pro načtení již existujícího projektu. Zde se okopíroval vstupní dialog z verze *PACK*, který jsem modifikoval pro uvedené potřeby.



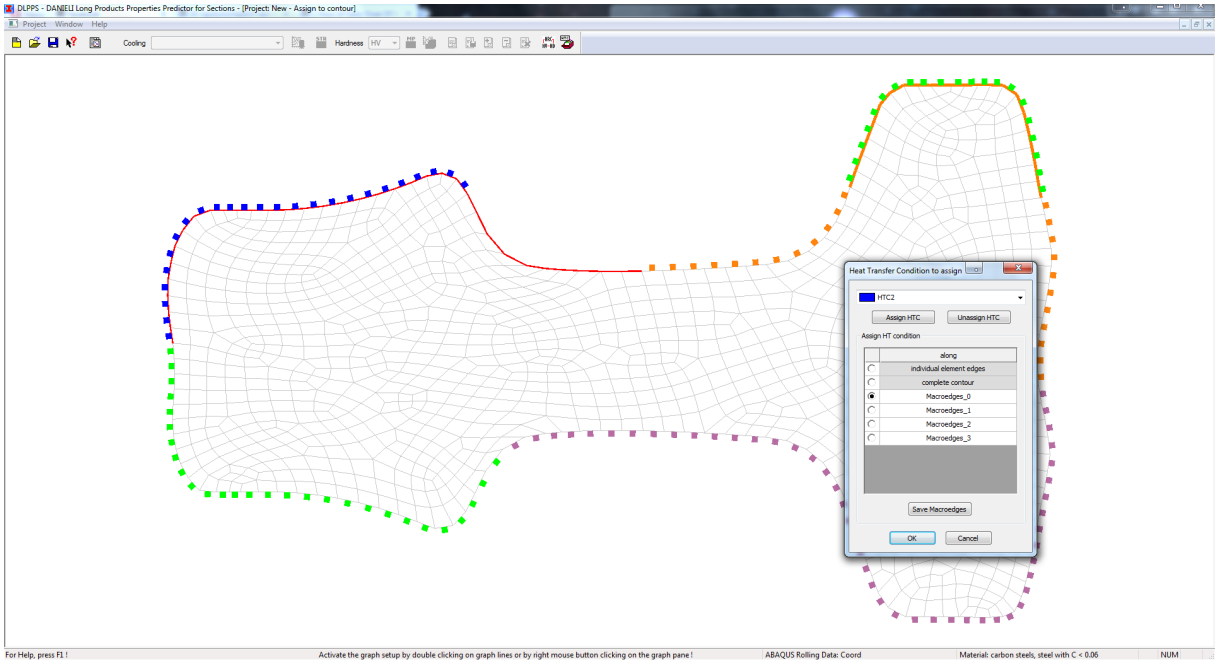
Obrázek 7: V levé části obrázku se nachází dialog s již konvertovanými soubory a v pravé části je spouštěcí dialog

4.2.4 Přidávání ochlazovacích podmínek na vnější hrany profilu

Tento úkol jsme rozdělili na dvě části. Kolega vytvářel nemodální dialog, který jsem křížovým odkazem propojil s modálním dialogem, jenž jsem vytvářel já. Nemodální dialog má metody pro vracení aktuální zvolené ochlazovací podmínky na sekci a zvolené pole makro hran ze seznamu. Také se v něm nachází tlačítka pro uložení makro hran a pro uložení, či zrušení změn a zavření obou dialogů. Mou částí práce bylo, aby uživatel mohl přiřazovat ochlazovací podmínky na obrys tělesa. Při pohybu myší se zvýrazní nejbližší hrana a pokud budou vytvořeny makro hrany, tak se zvýrazní celá nejbližší makro hrana. Při výběru je pomocí tlačítka control možnost vybírat více hran, nebo na již vybrané hraně zrušit výběr. Dále je možno při kliku a tahu myši, vybrat všechny hrany, či makro hrany, uvnitř výběrového obdélníku. Taktéž je možno výběr hran rozšiřovat, či zmenšovat pomocí tlačítka control a taženého výběru myší zároveň.

Pro možnost výběru obrysových hran, bylo zapotřebí je prvně vyhledat, na což jsem použil již vytvořenou třídu v *PACK* verzi, která vytvářela objekt s popisem obrysu. Tato data jsem vložil do nové struktury, kde jsem potřeboval vědět jednotlivé souřadnice bodů, lokální indexy obrysových hran, přiřazenou podmínku a úhel, kterým hrana směřovala. Pomocí těchto hodnot jsem byl schopen vypočíst, která hrana je v uživatelem definované vzdálenosti od obrysu nejblíže a hranu poté zvýraznit. Abych však mohl hranu zvýraznit, musel jsem vytvořit třídu, jenž dědila z manažeru knihovny ContourLib a v samotné knihovně přidat dvě virtuální metody, kdy první se spouští na konci vykreslování a druhá na začátku. Díky těmto metodám, a funkcím z OpenGL, jsem byl schopen přes dědění kreslit zvýrazněné a vybrané hrany zvlášť. Pokud byla na hranu přiřazena podmínka, kreslí se čtverec přiléhající na vnější stranu hrany, je také otočený ve směru hrany a má barvu přiřazené ochlazovací podmínky. Dle přiřazených podmínek na po sobě jdoucích hranách a vybraných hran, které mají přednost, se vytváří po stisknutí

tlačítka makro hrany. Při zavírání dialogů se stávalo, že nemodální dialog se správně neukončil a opětovné spuštění tak bylo chybné. Vytvořil jsem tedy statický ukazatel, na nemodálním dialog. Ten při inicializaci, pokud nebyl prázdný, smazal objekt ukazatele a uložil do něj novou adresu.



Obrázek 8: Zobrazení výběru hran a přiřazení ochlazovacích podmínek

- *V levé horní části profilu jsou hrany překryty aktivním výběrem červeně.*
- *V pravé horní části profilu jsou hrany překryty oranžovým zvýrazněním, jakožto vysvícení nejbližší makro hrany ke kurzoru.*
- *Jednotlivé barevné čtverečky označují přiřazenou ochlazovací podmínku. Hrany bez čtverečků jsou hrany bez přestupu tepla, a tedy nebudou ochlazovány.*

```
COLORREF crNodes;
ContourSpace::CMeshNode *pMeshNode;
int nNodes = GetChecker()->GETLISTS(1);

double CubeSize = (xSize*1.5)*SymbolSize;

glBegin( GL_QUADS );
for ( int i = 0; i < m_nodes->GetCountMeshNodes(); i++)
{
    if (MacroData[0][i].Assigned == TRUE)
    {
        pMeshNode = m_nodes->GetMeshNode(i);
        crNodes = pMeshNode->GetColor();
        glColor3d(GETR(crNodes), GETG(crNodes), GETB(crNodes));

        double Fi = m_AnglesToTranslateNodes[i].first;
        double Ni = m_AnglesToTranslateNodes[i].second;

        double x0 = pMeshNode->GetX();
        double y0 = pMeshNode->GetY();

        glVertex3d((x0+(CubeSize*cos(Fi))), (y0+(CubeSize*sin(Fi))), 0.0);
        glVertex3d((x0+(CubeSize*cos(Fi+PI))), (y0+(CubeSize*sin(Fi+PI))), 0.0);

        double x1 = x0+(CubeSize*cos(Ni)*2);
        double y1 = y0+(CubeSize*sin(Ni)*2);

        glVertex3d(x1+(CubeSize*cos(Fi+PI)), y1+(CubeSize*sin(Fi+PI)), 0.0);
        glVertex3d((x1+(CubeSize*cos(Fi))), (y1+(CubeSize*sin(Fi))), 0.0);
    }
}
glEnd();
glEndList();
```

Výpis 4: Vykreslení čtverců pomocí OpenGL, jakožto ochlazovací podmínky, na vnější hraně profilu

4.2.5 Pole technologií

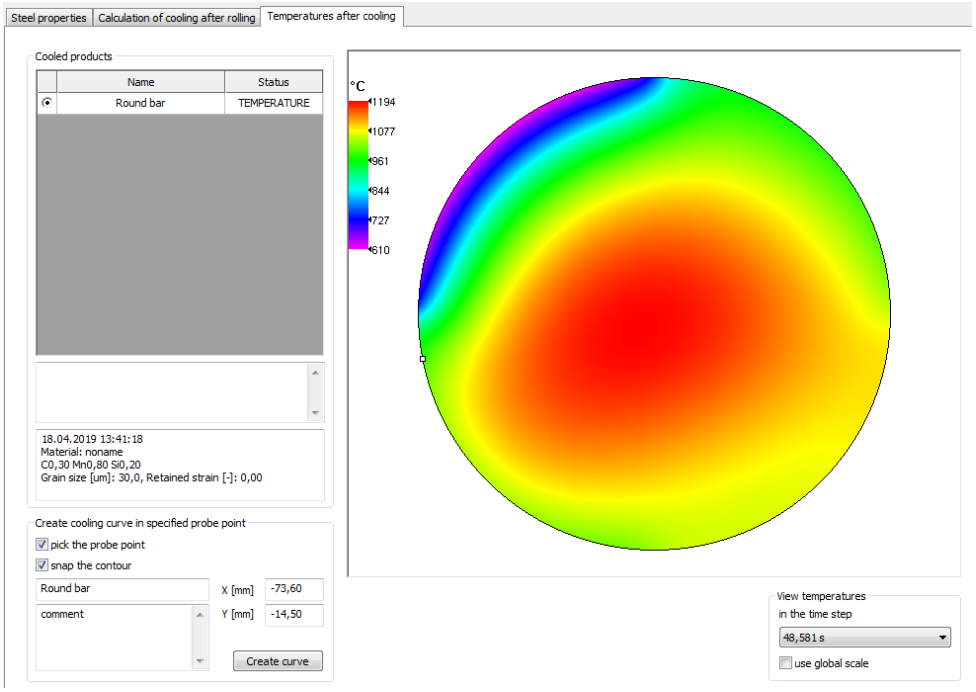
Následovalo vytvoření dynamického pole technologií, ve kterém jsou uloženy sekce, jejich podmínky a přiřazené podmínky pro jednotlivé sekce na tělese. Tyto technologie je možno přejmenovat, duplikovat, vytvářet a mazat. Původně jsem vytvářel změnu technologie, jako uložení dat z dialogu do technologie a následně načtení dat z technologie do dialogu. To vše pomocí kopírování objektů, což je časově náročné a vznikalo mnoho chyb. Přepsal jsem tedy dialog, aby používal ukazatele. Nyní při změně technologie, stačí změnit ukazatele na technologii, čímž se eliminovalo značné množství chyb a celý proces se urychlil.

4.2.6 Zobrazení výsledků a rozšíření funkcionality

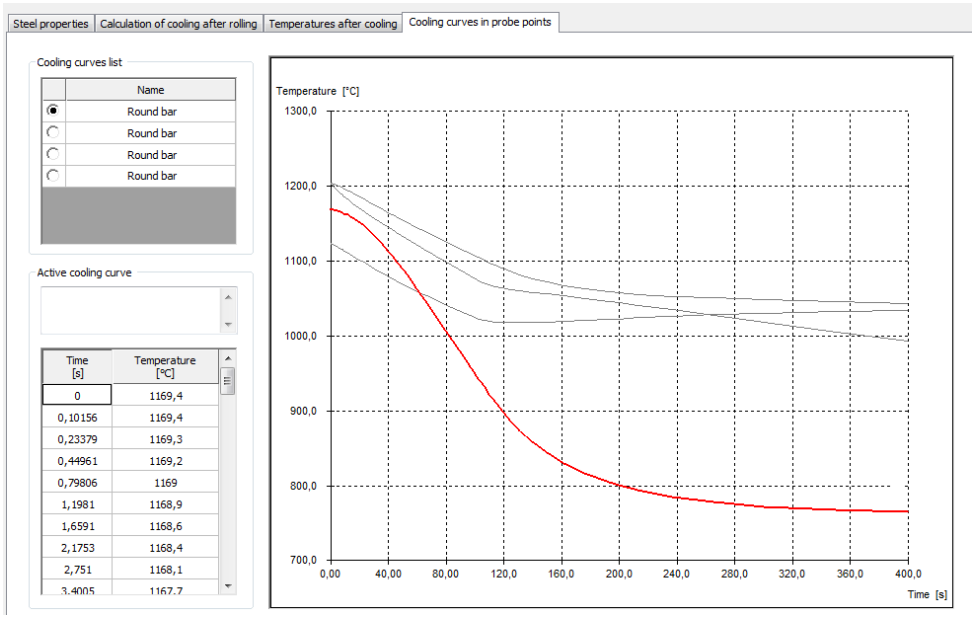
Všechny třídy a dialog patřící k zobrazování a zvolení bodu, na vytvoření křivky, jsem zkopíroval z verze *PACK* a upravil je tak, aby počítaly pouze s jednotlivými tělesy a ne s polem těles, jako je tomu potřeba v *PACK* verzi. Tento dialog jsem rozšířil o přichytávání výběrčího bodu k okraji, pro vytváření ochlazovacích křivek přesně na povrchu tělesa. Opět jsem tedy využil, již vytvořené třídy pro vytvoření obrysu tělesa, a jako v dialogu pro vybírání hran a přiřazování podmínek, jsem počítal nejbližší hranu. Jelikož hrany mohou být různě dlouhé a uživatel si může graf přiblížit a chtít vytvořit bod kdekoliv na hraně, vytvořil jsem metodu, která počítá nejbližší bod, na již zjištěné hraně, k bodu ukazatele myši. Výpočet jsem provedl pomocí půlicího algoritmu. Vypočetl jsem středový bod mezi body hrany a počítal, která z dvojic bodů, je blíže k bodu myši. Dva nejbližší body jsem si zvolil jako hlavní a opět jsem vypočetl středový bod a takto algoritmus opakoval, dokud nebyla provedena 12. iterace tohoto algoritmu. Tím jsem získal maximální odchylku 0.024% celkové délky hrany. Přichytávací bod, se zobrazuje jako malý čtverec a klasický bod bez přichycení se zobrazuje jako nitkový kříž. Tato rozšiřující funkčnost byla přidána i do verze *PACK*.

4.3 Verze *BTMP*

Jako v předchozích popisovaných verzích, bylo nejdříve potřeba vytvořit novou definici verze a vše k ní potřebné. Díky tomu, že tento program je podobný k verzím *PACK* a *DLPPS*, bylo možné většinu dialogů pouze nakopírovat a upravit, popřípadě třídy a dialogy z *PACK* verze přeložit do angličtiny. Tato verze používá pouze dva profily, identické s profily ve verzi *PACK*. U těchto profilů bylo potřeba upravit výpočet koeficientu škálování profilů, jelikož byly vytvořeny nové soubory s menší bodovou sítí. Dialog s vlastnostmi ocelí je použit z verze *DLPPS*. Dialog s výpočtem ochlazování je přeložen do angličtiny z české verze *PACK* a ochlazovací podmínky jsou z verze *DLPPS*. Pro zobrazení výsledků a vytváření ochlazovacích křivek, jsou použity třídy a dialogy pro verzi *DLPPS*, v nichž nebylo třeba nic upravovat, jelikož funkčnost má být identická. V programu *BTMP* nebylo použito přiřazování podmínek na těleso uživatelem, ale rozdělení na 4 souměrné strany. Zablokoval jsem tedy možnost vytváření, nebo mazání ochlazovacích podmínek a zachoval pouze vytváření a mazání sekcí. V každé sekci tedy jsou jen 4 podmínky.



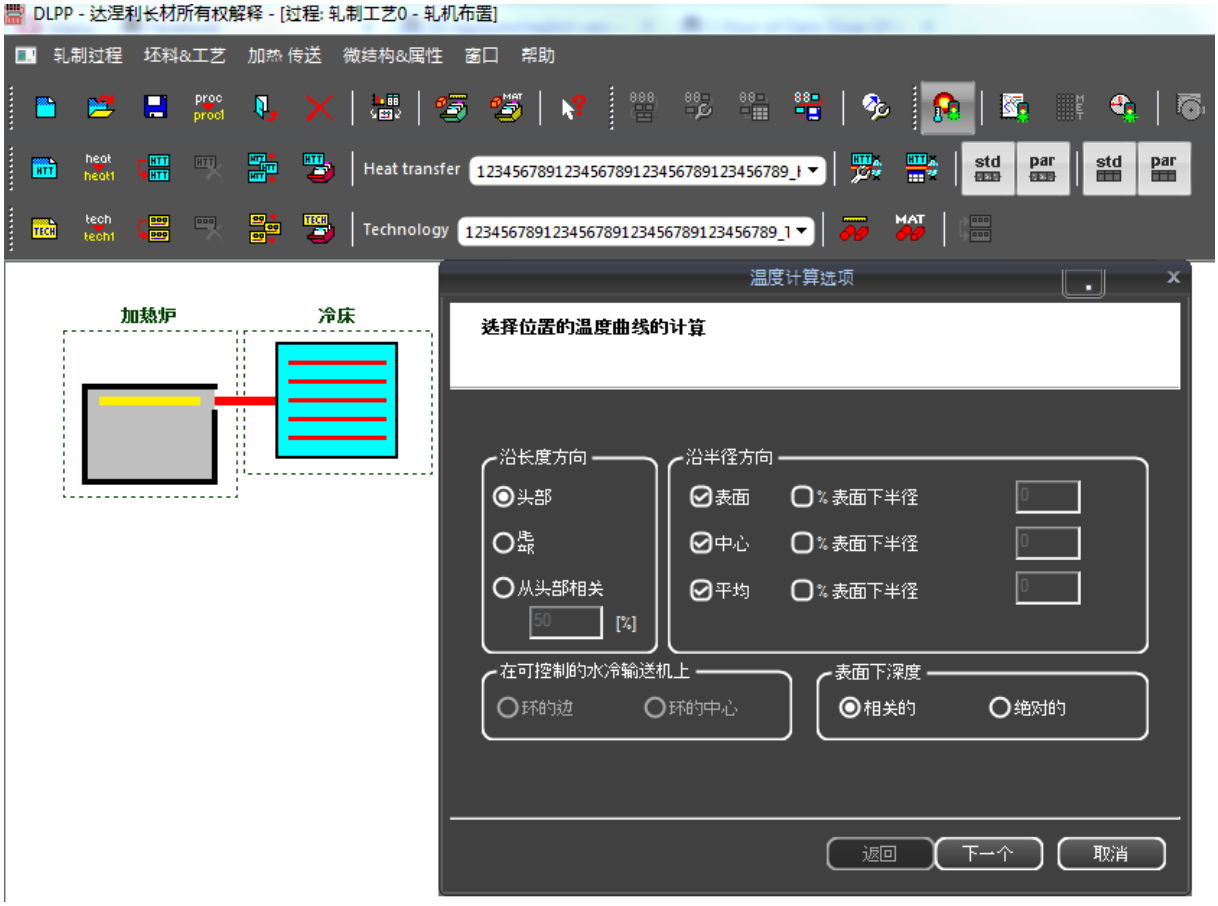
Obrázek 9: Výsledek ochlazovací simulace verze BTMP s vyznačeným krajním bodem v levé části profilu



Obrázek 10: Zobrazení vytvořených 1D ochlazovacích křivek

4.4 Čínská verze programu DLPP

Zde jsme vytvořili release čínské verze, která je v čínštině, a obnovili instalační soubory. Při kontrole programu, se vyskytly problémy s překladem, kdy menu a několik dalších názvů bylo v angličtině. Po zdlouhavém hledání možné příčiny v nastavení projektu, znakové sady a mnoho dalších možností, jsme objevili zdroj chyby. Chyba vznikala v registrech, kam si MFC ukládá pozice toolbarů a názvy v menu. Chyba se tedy děla jen na firemních počítačích, kde jsou nainstalované i anglické verze a čínskému uživateli by se tak tato chyba nevyskytla. Některé názvy i přesto byly pořád v angličtině, ovšem to bylo způsobeno rozšiřováním funkcionalit v programu a nepřidáním textových řetězců do čínských zdrojových textů, což jsme opravili přidáním textů pomocí internetového překladače.



Obrázek 11: Vzhled čínské verze

4.5 Italský překlad programu DLPP

Vznikl požadavek přeložení programu DLPP do italštiny. Bylo tedy potřeba rozšířit stávající zdrojové soubory o italštinu. Ve firmě již byl na toto vytvořen program v jazyce C#, když se program překládal do čínštiny. Tento program tak bylo potřeba upravit, aby pracoval na části definované pro italský jazyk, kterou jsme museli ve zdrojových souborech vytvořit ručně. Jelikož jsme byli během tohoto požadavku zaneprázdnění, byl poslán překladateli původní textový anglický soubor, vytvořený pro překlad do čínské verze, který byl však zastaralý a neobsahoval tak všechny textové řetězce. Po přeložení verze do italštiny většina názvů byla správná, avšak názvy dialogů a group boxů byly většinou nesprávné. Museli jsme tedy projít všechny dialogy a jednotlivé názvy kontrolovat s anglickou verzí a opravovat je. Jelikož se názvy v programu často opakují, nebylo potřeba posílat příliš mnoho textů, na opětovný překlad, překladateli. Během procházení dialogů jsme také museli zvětšovat některá statická textová pole, jelikož italské názvy jsou mnohem delší, než anglické, a nezobrazovaly se tak celé.

4.6 Pomocné třídy

Během mé praxe jsme založili několik tříd, které usnadňují práci ve firmě. Mezi jedny z hlavních patří třída CLog, díky které rychle vytváříme přehledné logy programu. Druhá třída, kterou bych rád zmínil, je třída Utils, kam vkládáme statické metody, které provádí často opakované úkony. Například převod z CStringu na string, načtení CSV souboru do vektorových polí, rekurzivní hledání všech souborů v uvedeném adresáři, výpočet úhlu od bodu k bodu, či vytvoření náhodné barvy.

```
double Utils::GetAngle(double XX, double XY, double YX, double YY){
    double angle = atan2(XY - YY, XX - YX);
    return angle = 180 + (angle * 180 / PI);
}

string Utils::CStringToString(CString text)
{
    return (CT2A(text.GetString()));
}

void Utils::ReadCSV(std::vector<std::vector<double>>& data, CString FilePath, char Delimiter, int
    Columns){

    std::vector<double> a;
    for (int i = 0; i < Columns; i++)
        data.push_back(a);

    std::ifstream file(FilePath);
    std::string str;
    int j = -1;

    while (std::getline(file, str, '\n'))
    {
        stringstream cell(str);
        string scell;
        while (std::getline(cell, scell, Delimiter))
        {
            j++;
            data[j].push_back(std::stod(scell));
            if (j == Columns-1)
            {
                j = -1;
            }
        }
    }
}
```

Výpis 5: Ukázka statických metod z třídy Utils

5 Závěr

5.1 Teoretické a praktické znalosti uplatněné v průběhu praxe

Během této praxe jsem uplatnil své znalosti C++, OpenGL a algoritmizace. Tyto znalosti jsem nabyl na střední a vysoké škole, na VŠ přesněji v předmětech Programování I, Programování II, dále v Algoritmy I, Algoritmy II a také v předmětu Základy počítačové grafiky, kde jsem se naučil pracovat s OpenGL, dědičností tříd a používat návrhové vzory. Řešení mnoha problémů jsem si ulehčil vytvářením vývojových diagramů, které jsem se naučil v předmětu Úvod do softwarového inženýrství.

5.2 Teoretické a praktické znalosti scházející v průběhu praxe

Nejvíce mi scházely znalosti o MFC a jak s ním pracovat. Dále také znalosti ohledně vytváření instalačních souborů, zkušenosti s verzováním, ať už pomocí webové služby GitHub, nebo programu TortoiseSVN a znalosti spojené s nastavením vlastností projektu ve Visual Studiu.

5.3 Celkové shrnutí individuální praxe

V úvodu této práce jsem popsal firmu, ve které jsem vykonával bakalářskou praxi. Dále popisuji svou pracovní pozici, projekty na nichž jsem pracoval a mé hlavní zaměření, jenž mi bylo přiděleno v týmu. Následně popisuji všechny technologie, které se ve firmě používají. Následuje popis firemních programů, které jsem v rámci této praxe rozšiřoval a upravoval. V další části jsou vypsány jednotlivé úkoly, jenž byly zadány týmu, ve kterém jsem pracoval, a jejich následné vypracování.

Na začátku byla má práce velmi pomalá a neefektivní, jelikož jsem se musel naučit pracovat s MFC a orientovat se ve velkém množství tříd a knihoven. Postupem času, kdy jsem pracoval na programu QTSteel a jeho knihovnách, jsem začínal chápat celkové propojení tříd a má práce se tak výrazně zefektivnila. Při občasných pracích na programu DLPP ovšem nejsem tak efektivní, jelikož je tento program psán jiným stylem, a vyhledávání různých metod a funkcí je tak zdlouhavější.

Během této praxe jsem hlavně nabyl zkušeností z hlediska programování od kolegů v týmu a naučil se také od nich několik firemních pravidel, které výrazně zpřehledňují kód a urychlují práci. Nejtěžší a zároveň nejzábavnější částí mé práce bylo, vytváření algoritmů a atomizace různých problémů.

Literatura

[1] ITA spol. s r.o. [online]. Copyright © 2007 [cit. 18.04.2019]. Dostupné z: http://www.ita-tech.cz/cs/

[2] cplusplus.com - The C++ Resources Network. [online]. Copyright © cplusplus.com, 2000 [cit. 23.04.2019]. Dostupné z: http://www.cplusplus.com/

[3] MFC Class Library Overview [online]. [cit. 14.04.2019]. Dostupné z: https://docs.microsoft.com/cs-cz/cpp/mfc/class-library-overview

[4] MFC Desktop Applications [online]. [cit. 14.04.2019]. Dostupné z: https://msdn.microsoft.com/en-us/library/d06h2x6e.aspx

[5] TortoiseSVN [online]. [cit. 15.04.2019]. Dostupné z: https://tortoisesvn.net/support.html

[6] ISTool [online]. [cit. 15.04.2019]. Dostupné z: http://www.istool.net/

[7] STROUSTRUP, Bjarne. C++ programovací jazyk. Praha: Softwarové Aplikace a Systémy, c1997. ISBN 80-901507-2-1.

[8] SEDGEWICK, Robert. Algorithms in C++. 3rd Edition. Boston, Massachusetts, USA: Addison-Wesley Professional, 1998. ISBN 978-0201350883.

[9] JONES, Richard M. Introduction to MFC programming with Visual C++. Upper Saddle River, N.J.: Prentice Hall PTR, c2000. ISBN 0130166294.

[10] ČADA, Ondřej. Objektové programování: naučte se pravidla objektového myšlení. Praha: Grada, 2009. Průvodce (Grada). ISBN 978-80-247-2745-5.